

Een belangrijk onderdeel binnen het OO programmeren zijn de object functies. Deze functies bepalen het gedrag dat een object vertoont en geven de mogelijkheid om objecten met elkaar te laten samenwerken. Via deze functies kunnen objecten met elkaar informatie uitwisselen en elkaars status (waarden van de properties) aanpassen. Verder bieden functies ook nog een ander voordeel: code die je mogelijk meerdere keren in de loop van je programma wil uitvoeren zijn gegroepeerd en op te roepen via de functienaam.

Oefening 1

Maak een blackjack programma dat de gebruiker toelaat om een inzet te plaatsen en kaarten te trekken. Wanneer het spel is beëindigd zal de gebruiker de mogelijkheid hebben een nieuw spel te starten. Hieronder een overzicht van de te gebruiken klassen:

Kaart

Properties

- Waarde (int): geeft de waarde terug van de kaart. Een andere klasse kan deze waarde niet aanpassen.
- Omschrijving (string): afhankelijk van de Waarde property zal een omschrijving worden teruggegeven 1 -> 10 / boer(11) / dame(12) / heer (13)

Met de onderstaande regel code kan je de waarde van je kaart bepalen (rechtstreekse initialisatie van de property of via de constructor):

```
Waarde= new Random().Next(1, 14);
```

Blackjack

Properties

- TotaalKaarten (int): bevat het huidige totaal van de getrokken kaarten. Kan enkel uitgelezen worden.
- Inzet (double): bevat de geplaatste inzet.
- SpelStatus (string): de huidige status van het spel: 0 = verloren / 1 = spel gestart / 2 = gewonnen

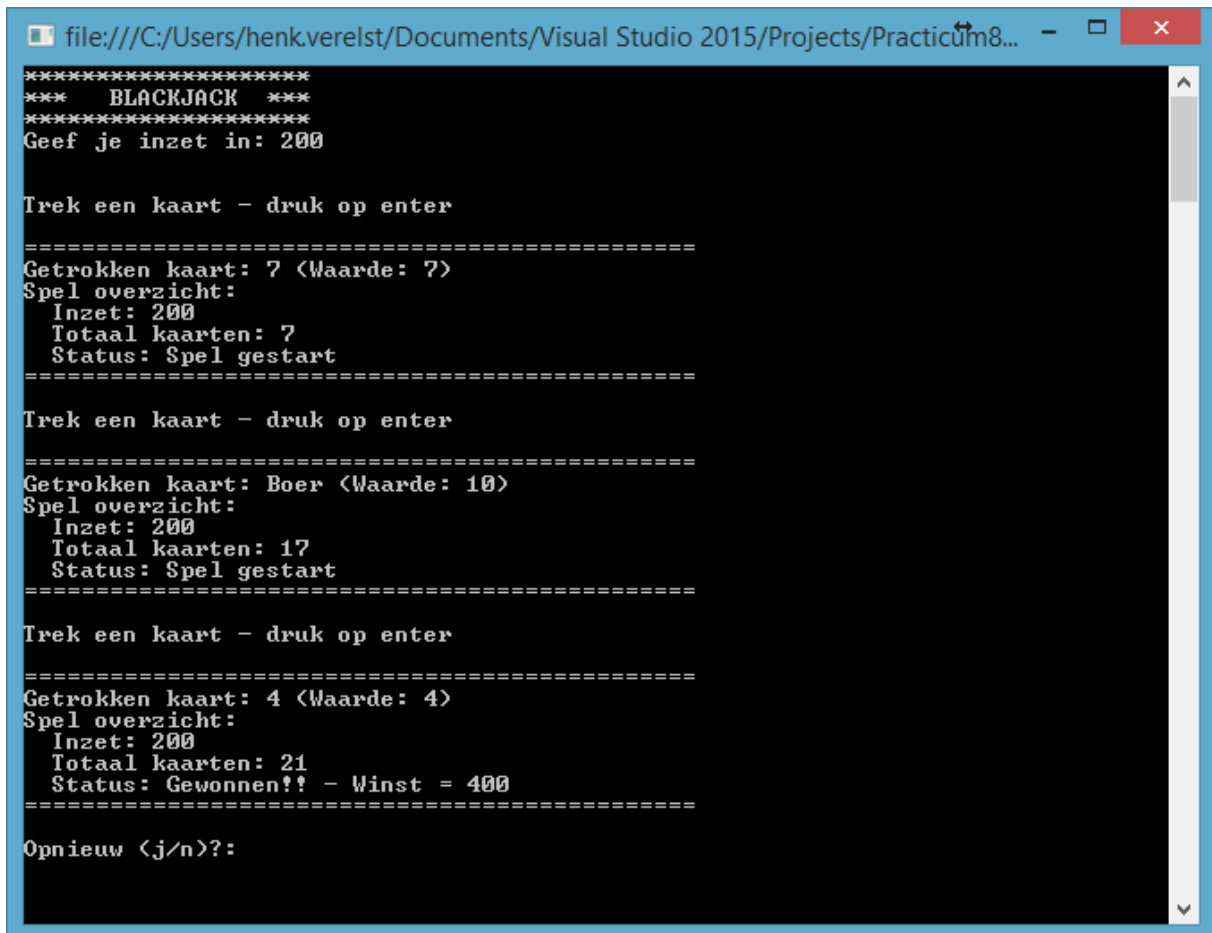
Functies

- TrekKaart: deze functie zal een nieuwe kaart aanmaken en de info teruggeven aan de functie caller
 - o Return (string): de omschrijving en waarde van de kaart
 - o Parameters: geen
- Inzetten: via deze functie zal de inzet van het spel kunnen bepaald worden.
 - o Return (bool): true indien inzet ok / false indien inzet negatief
 - o Parameters:
 - inzet (double): de waarde van de inzet

- GeefOverzicht: geeft een overzicht van het huidige spel (zie hieronder voor screenshots). Een spel dat verloren is zal het verlies aangeven (= inzet). Bij winst win je 2 x je inzet.
 - o Return (string): een overzicht van de bovenstaande properties
 - o Parameters: geen

Hieronder een mogelijk spelverloop:

1. Blackjack - Gewonnen



```
file:///C:/Users/henk.verelst/Documents/Visual Studio 2015/Projects/Practicum8... - [X]
*****
***  BLACKJACK  ***
*****
Geef je inzet in: 200

Trek een kaart - druk op enter
=====
Getrokken kaart: 7 (Waarde: 7)
Spel overzicht:
  Inzet: 200
  Totaal kaarten: 7
  Status: Spel gestart
=====

Trek een kaart - druk op enter
=====
Getrokken kaart: Boer (Waarde: 10)
Spel overzicht:
  Inzet: 200
  Totaal kaarten: 17
  Status: Spel gestart
=====

Trek een kaart - druk op enter
=====
Getrokken kaart: 4 (Waarde: 4)
Spel overzicht:
  Inzet: 200
  Totaal kaarten: 21
  Status: Gewonnen!! - Winst = 400
=====

Opnieuw <j/n>?:
```

2. Blackjack - Verloren

```
file:///C:/Users/henk.verelst/Documents/Visual Studio 2015/Projects/Practicum8... - [X]
*****
***  BLACKJACK  ***
*****
Geef je inzet in: 200

Trek een kaart - druk op enter

=====
Getrokken kaart: 7 (Waarde: 7)
Spel overzicht:
  Inzet: 200
  Totaal kaarten: 7
  Status: Spel gestart
=====

Trek een kaart - druk op enter

=====
Getrokken kaart: Dame (Waarde: 11)
Spel overzicht:
  Inzet: 200
  Totaal kaarten: 18
  Status: Spel gestart
=====

Trek een kaart - druk op enter

=====
Getrokken kaart: Boer (Waarde: 10)
Spel overzicht:
  Inzet: 200
  Totaal kaarten: 28
  Status: Verloren - Verlies = 200
=====

Opnieuw (j/n)? : _
```

3. Wanneer een ongeldige inzet wordt ingegeven zal opnieuw gevraagd worden een inzet in te geven

```
file:///C:/Users/henk.verelst/Documents/Visual Studio 2015/Projects/Practicum8... - [X]
*****
***  BLACKJACK  ***
*****
Geef je inzet in: -444
Gelieve een positief getal in te geven
Geef je inzet in:
```